PCS

Periphere
Computer Systeme GmbH

Unser Zeichen 400-sk München, den 1.12.83

PCS GmbH, Pfälzer-Wald-Str. 36, 8000 München 90

Techn. Hoogeschool Twente
Abt. E.F.
Herrn Höfgen
Postfach 217

7500AE ENSCHEDE
NIEDERLANDE

Sie erhalten mit diesem Begleitschein

☒ folgende Unterlagen:

Sources der Bit-Map-Terminal
Demoprogramme

☐ mit bestem Dank zurück

☐ mit der Bitte um Stellungnahme

☐ zu Ihrer/unserer Nachricht vom

☒ zur Kenntnisnahme

☐ mit der Bitte um Erledigung bis

mit freundlichen Grüßen PCS GmbH

i.A. Stephan Spath, Produktmanager

```
/*
        Standalone bouncing square demo
        */
#include "legsdef.h"
#define  Xmax  (Bwide-1)
#define  Ymax  (Bhi-1)

        long  valof(),  ndelay = 1000;
#define  delay( n )  { long t = n;  do {} while( --t >= 0 ); }


main( nargs, args )  char* args[];
{
        if( args[1])  ndelay = valof( args[1]);
        BouncingSquare();
        }

BouncingSquare()
 {
    int x, y, vx, vy;

    GXfunction = GXset;
    ras_write( 1023,1023,0,0);
    x=0;
    y=0;
    vx=4;
    vy=0;
    GXfunction = GXinvert;
    while (1) {
        ras_write( 20, 20, x, y );
        x=x+vx;
        if (x>(Xmax-20))
         {
                /*
                 * Bounce off the right edge
                 */
            x=2*(Xmax-20)-x;
            vx= -vx;
         }
        else if (x<0)
          {
                /*
                 * bounce off the right edge
                 */
            x= -x;
            vx= -vx;
         }
        vy=vy+1;
        y=y+vy;
        if (y>=(Ymax-20))
         {
            y=Ymax-20;
            if (vy<20) vy=1-vy;
            else vy=vy / 20 - vy;
            if (vy==0)
             {
                x=0;
                y=0;
                vx=4;
                vy=0;
             }
         }
    delay( ndelay );
    }
 }
```

```
/*  ell.c:  draw circles / ellipses  */
#include "legsdef.h"
#include "../bz.h"
#define  x0  (Bwide / 2)
#define  y0  (Bhi   / 3)
        long  valof(),  ndelay = 1000;
#define  delay( n )  { long t = n;  do {} while( --t >= 0 ); }


main( nargs, args )  char* args[];
{       reg int  rad, n, op, ellipse = 2;
        extern int Bcircleshift;
        if( args[1] ){ ellipse      = valof( args[1]);
        if( args[2] )  Bcircleshift = valof( args[2]);
        }
        Bfill( 0, 0, Bhi, Bwide, Bblack );

    for( n = 0;  ++n; ){
        op = (n & 1 ?  Bwhite  : Bblack );

        for( rad = 50;  rad < x0;  rad += 10 ){
                if( ellipse == 0  && rad >= y0 )  break;
/*.....................................................................*/
                Bcircle( x0, y0, rad, op, ellipse );
                delay( ndelay );
        }
                delay( 100 * ndelay );
    }
}
```

```
/*  Fputc( c, x, y ):   char from Berkeley font  ->  bip
                global:  Font  Fputc_op  */
*/
#include <bip/ops.h>
#include "font.h"       /*  <vfont.h> +  */
#include "../bz.h"

struct font*  Font; /*  current font  from Fontread  */
short  Fputc_op = Blblack;


Fputc( c, x, y )  reg char c;
{       reg letterty *l = & Font->letters[ c ];
        if( l->nbytes == 0 ){
                if( In( c,  'a', 'z' ))  c += 'A'-'a';
                else  c = '?';
                l = & Font->letters[ c ];
                if( l->nbytes == 0 )  return;
        }
/*.....................................................................*/
        Bput16( x - l->left,  y - l->up,
                l->up + l->down,  l->left + l->right,
                & Font->bits[ l->addr ],
                Fputc_op );
}
```

```
/*  sun -> bip  demos  */
#include <bip/ops.h>

extern short
        GXfunction,
        GXwidth,
        GXcontrol;

#define  GXset      Bwhite
#define  GXclear    Bblack
#define  GXinvert   Binvert
#define  GXxor      (Bsource ^ Bdest)
#define  GXvideoEnable  0
#define  true  1

#define  ras_copy   Blt
/*  #def ras_write( h w x y )   Bfill( x y h w GXfunction )  */
```

```
/*
        Munching squares demo
        */
#       include "legsdef.h"
#define  Ymax  799
#define  param  12345
main()
{
        if( GXfind()) exit(1);
        GXcontrol = GXvideoEnable;
        munch();
        }


munch() {
    int x, y, t, dt;
    GXfunction = GXset;
    ras_write( 1023,1023,0,0);
    if (param==0) dt=1; else dt=param;
    t=0;
    GXfunction = GXinvert;
    while (true) {
        for (x=0; x<=511; x++) {
            y=(511 & (x ^ t));
            if( y+y < Ymax )
            ras_write (2, 2, 2*x, 2*y);
        }
        t=t+dt;
    }
}
```

```
/*  ray.c  triv lines from centre  */
#include "legsdef.h"
#include "../bz.h"

#define  x0  ((Bwide / 3)  & -32)
#define  y0  ((Bhi   / 3)  & -32)


main( nargs, args )  char* args[];
{       reg int  x, y, n, op,  sp = 16,  nsleep = 0;
        if( args[1] ){  sp      = valof( args[1]);
        if( args[2] )   nsleep  = valof( args[2]);
        }
        Bfill( 0, 0, Bhi, Bwide, Bwhite );

  for( n=0;  ++n; ){
    op = (n & 1  ? Bblack  : Bwhite);
    for( x = 0;  x < Bwide;  x += sp )     Bline( x0, y0, x,       0,    op );
    for( y = 0;  y < Bhi;    y += sp )     Bline( x0, y0, Bwide-1, y,    op );
    for( x = Bwide-1;  x >= 0;  x -= sp )  Bline( x0, y0, x,       Bhi-1, op );
    for( y = Bhi-1;    y >= 0;  y -= sp )  Bline( x0, y0, 0,       y,    op );
    sleep( nsleep );
  }
}
```

```
/*  recursive tree
        after Wyvill,  June Software P&E
 */
#include "traline.h"
#include <bip/ops.h>
#include "../bz.h"

        int  treemax = 8;
        trans  A, B, C;
        struct line  unitl = { 0., 0.,  0., 1. };

main( nargs, args )  char* args[];
{
        if( args[1])  treemax = valof( args[1]);
        Bfill( 0, 0, Bhi, Bwide,  Bwhite );
        A = make_trans(  30., 0.75,  0., 1./3  );
        B = make_trans( -35., 0.6,   0., 2./3  );
        C = make_trans(  30., 0.5,   0., 1.    );

        tree( &unitl, 0 );
}

/*.....................................................................*/
tree( l, n )  reg line l;
{       struct line  Al, Bl, Cl;
        drawline( l );
        if( ++n >= treemax
        || (l->p.x == l->q.x  && l->p.y == l->q.y))
                return;
        traline( A, l, &Al );   tree( &Al, n );
        traline( B, l, &Bl );   tree( &Bl, n );
        traline( C, l, &Cl );   tree( &Cl, n );
}
```

```
/*  mouse 16*16 big blocks  -->  wallpaper  */

#include <bip/ops.h>
#include "../bz.h"

bfillpat( x, y, h0, w0, pat )  int pat[16];
{       reg int h, w;
        if( h0 <= 0  || w0 <= 0 )
                return;
        Bput16( x, y, 16, 16, pat, Bsource );
        for( w=16;  w<w0;  w+=w )
                Bit( x, y, 16,  MIN( w, w0-w ),
                     x+w, y,  Bsource );
        for( h=16;  h<h0;  h+=h )
                Bit( x, y, MIN( h, h0-h ), w,
                     x, y+h,  Bsource );
}

bfillframe( x, y, h, w, pat )  int pat[16]; /*  fill frame around xyhw  */
{       bfillpat( 0,    0,      y,       Bwide, pat );
        bfillpat( 0,    y+h,    Bhi-y-h, Bwide, pat );
        bfillpat( 0,    y,      h,       x,     pat );
        bfillpat( x+w,  y,      h,       Bwide-x-w,  pat );
}


static int  arrow[] = {
        0x8000, 0xc000, 0xe000, 0xf000, 0xf800, 0xfc00, 0xfe00,
        0xf000, 0xd800, 0x9800,
        0x0c00, 0x0c00,  0x0600, 0x0600,  0x0300, 0x0300 };

main()
{       bfillframe( 320, 192,  256, 256,  arrow );
        /*  next:  mouse 16*16 big blocks  */
}
```